嘟嘟独立博客

爱生活爱编码

search...

文章目录 隐藏目录



# Spring Boot干货系列：（三）启动原理解析

Spring Boot干货系列 Spring Boot

关闭阅读模式

2017-03-09

## 前言

前面几章我们见识了SpringBoot为我们做的自动配置，确实方便快捷，但是对于新手来说，如果不大懂SpringBoot内部启动原理，以后难免会吃亏。所以这次博主就跟你们一起一步步揭开SpringBoot的神秘面纱，让它不在神秘。

## 正文

我们开发任何一个Spring Boot项目，都会用到如下的启动类

```
@SpringBootApplication
public class Application {
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

从上面代码可以看出，Annotation定义（@SpringBootApplication）和类定义（SpringApplication.run）最为耀眼，所以要揭开SpringBoot的神秘面纱，我们要从这两位开始就可以了。

### —— SpringBootApplication背后的秘密 ——

```
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
```

查看 11 条评论

```
@Inherited
@SpringBootConfiguration
@EnableAutoConfiguration
@ComponentScan(excludeFilters = {
        @Filter(type = FilterType.CUSTOM, classes = TypeExcludeFilter.class),
        @Filter(type = FilterType.CUSTOM, classes = AutoConfigurationExcludeFilter.class) })
public @interface SpringBootApplication {
...
}
```

虽然定义使用了多个Annotation进行了原信息标注，但实际上重要的只有三个Annotation：

- @Configuration（@SpringBootConfiguration点开查看发现里面还是应用了@Configuration）
- @EnableAutoConfiguration
- @ComponentScan

所以，如果我们使用如下的SpringBoot启动类，整个SpringBoot应用依然可以与之前的启动类功能对等：

```
@Configuration
@EnableAutoConfiguration
@ComponentScan
public class Application {
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

每次写这3个比较累，所以写一个@SpringBootApplication方便点。接下来分别介绍这3个Annotation。

## —— @Configuration ——

这里的@Configuration对我们来说不陌生，它就是JavaConfig形式的Spring Ioc容器的配置类使用的那个@Configuration，SpringBoot社区推荐使用基于JavaConfig的配置形式，所以，这里的启动类标注了@Configuration之后，本身其实也是一个IoC容器的配置类。
举几个简单例子回顾下，XML跟config配置方式的区别：

- 表达形式层面

基于XML配置的方式是这样：

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.o
       default-lazy-init="true">
    <!--bean定义-->
</beans>
```

而基于JavaConfig的配置方式是这样：

```
@Configuration
public class MockConfiguration{
    //bean定义
}
```

任何一个标注了@Configuration的Java类定义都是一个JavaConfig配置类。

- 注册bean定义层面

基于XML的配置形式是这样：

```
<bean id="mockService" class="..MockServiceImpl">
    ...
</bean>
```

而基于JavaConfig的配置形式是这样的：

```
@Configuration
public class MockConfiguration{
    @Bean
    public MockService mockService(){
        return new MockServiceImpl();
    }
}
```

任何一个标注了@Bean的方法，其返回值将作为一个bean定义注册到Spring的IoC容器，方法名将默认成该bean定义的id。

- 表达依赖注入关系层面

为了表达bean与bean之间的依赖关系，在XML形式中一般是这样：

```
<bean id="mockService" class="..MockServiceImpl">
    <propery name ="dependencyService" ref="dependencyService" />
</bean>

<bean id="dependencyService" class="DependencyServiceImpl"></bean>
```

而基于JavaConfig的配置形式是这样的：

```
@Configuration
public class MockConfiguration{
    @Bean
    public MockService mockService(){
        return new MockServiceImpl(dependencyService());
    }

    @Bean
    public DependencyService dependencyService(){
        return new DependencyServiceImpl();
    }
}
```

如果一个bean的定义依赖其他bean,则直接调用对应的JavaConfig类中依赖bean的创建方法就可以了。

## —— @ComponentScan ——

@ComponentScan这个注解在Spring中很重要，它对应XML配置中的元素，@ComponentScan的功能其实就是自动扫描并加载符合条件的组件（比如@Component和@Repository等）或者bean定义，最终将这些bean定义加载到IoC容器中。

我们可以通过basePackages等属性来细粒度的定制@ComponentScan自动扫描的范围，如果不指定，则默认Spring框架实现会从声明@ComponentScan所在类的package进行扫描。

> 注：所以SpringBoot的启动类最好是放在root package下，因为默认不指定basePackages。

## —— @EnableAutoConfiguration ——

个人感觉@EnableAutoConfiguration这个Annotation最为重要，所以放在最后来解读，大家是否还记得Spring框架提供的各种名字为@Enable开头的Annotation定义？比如@EnableScheduling、@EnableCaching、@EnableMBeanExport等，@EnableAutoConfiguration的理念和做事方式其实一脉相承，简单概括一下就是，**借助@Import的支持，收集和注册特定场景相关的bean定义**。

- @EnableScheduling是通过@Import将Spring调度框架相关的bean定义都加载到IoC容器。
- @EnableMBeanExport是通过@Import将JMX相关的bean定义加载到IoC容器。

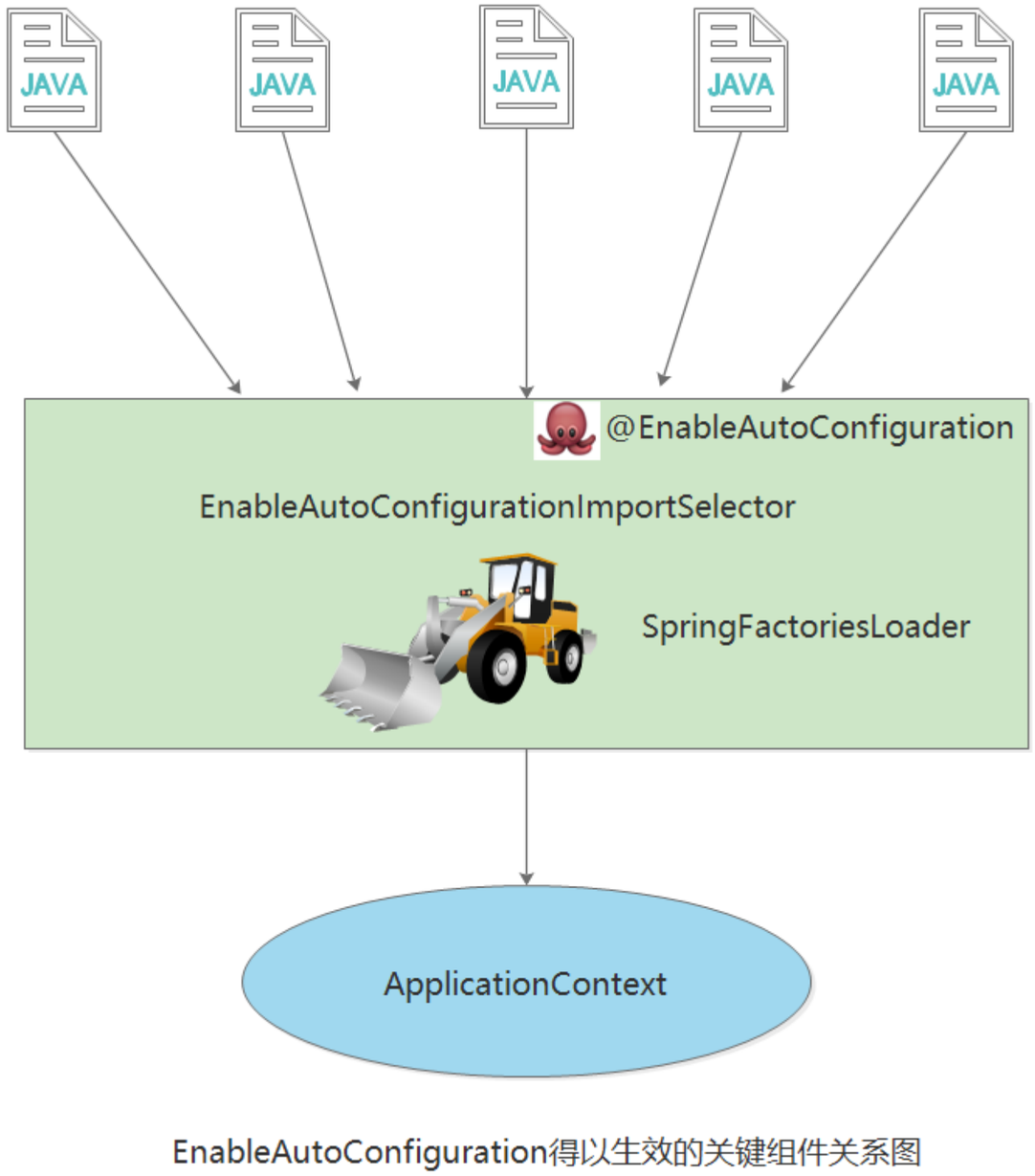
而@EnableAutoConfiguration也是借助@Import的帮助，将所有符合自动配置条件的bean定义加载到IoC容器，仅此而已！

@EnableAutoConfiguration作为一个复合Annotation,其自身定义关键信息如下：

```
@SuppressWarnings("deprecation")
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
@Documented
@Inherited
@AutoConfigurationPackage
@Import(EnableAutoConfigurationImportSelector.class)
public @interface EnableAutoConfiguration {
    ...
}
```

其中，最关键的要属@Import(EnableAutoConfigurationImportSelector.class)，借助EnableAutoConfigurationImportSelector，@EnableAutoConfiguration可以帮助SpringBoot应用将所有符合条件的@Configuration配置都加载到当前SpringBoot创建并使用的IoC容器。就像一只“八爪鱼”一样

借助于Spring框架原有的一个工具类：SpringFactoriesLoader的支持，@EnableAutoConfiguration可以智能的自动配置功效才得以大功告成！




EnableAutoConfiguration得以生效的关键组件关系图

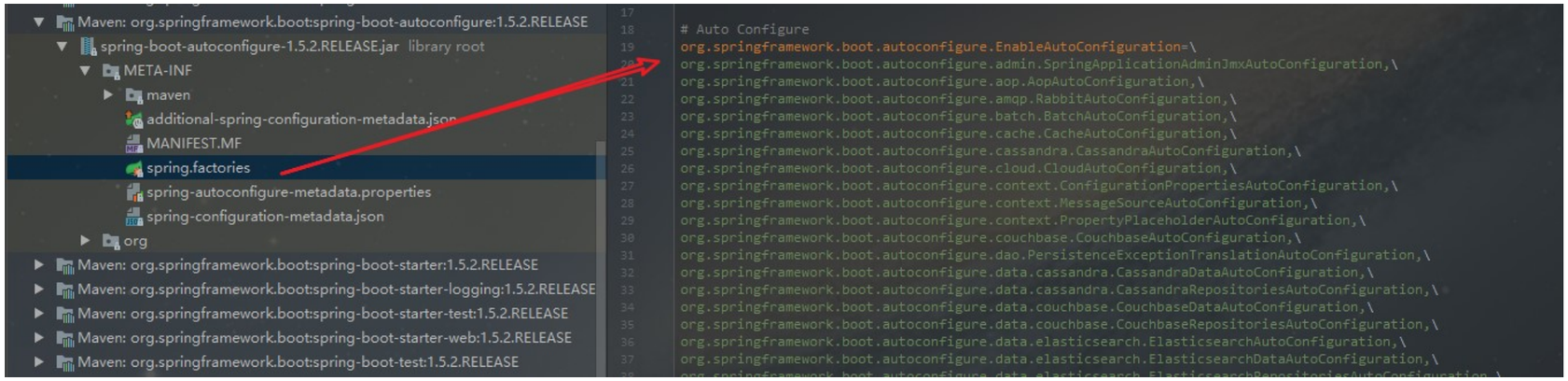
**自动配置幕后英雄：SpringFactoriesLoader详解**

SpringFactoriesLoader属于Spring框架私有的一种扩展方案，其主要功能就是从指定的配置文件META-INF/spring.factories加载配置。

```
1 public abstract class SpringFactoriesLoader {
2     //...
3     public static <T> List<T> loadFactories(Class<T> factoryClass, ClassLoader classLoader) {
4         ...
5     }
6
7
```

```
        ....
    }
}
```

配合@EnableAutoConfiguration使用的话，它更多是提供一种配置查找的功能支持，即根据@EnableAutoConfiguration的完整类名org.springframework.boot.autoconfigure.EnableAutoConfiguration作为查找的Key,获取对应的一组@Configuration类




上图就是从SpringBoot的autoconfigure依赖包中的META-INF/spring.factories配置文件中摘录的一段内容，可以很好地说明问题。

所以，@EnableAutoConfiguration自动配置的魔法骑士就变成了：**从classpath中搜寻所有的META-INF/spring.factories配置文件，并将其中org.springframework.boot.autoconfigure.EnableutoConfiguration对应的配置项通过反射（Java Refletion）实例化为对应的标注了@Configuration的JavaConfig形式的IoC容器配置类，然后汇总为一个并加载到IoC容器。**

## —— 深入探索SpringApplication执行流程 ——

SpringApplication的run方法的实现是我们本次旅程的主要线路，该方法的主要流程大体可以归纳如下：

1） 如果我们使用的是SpringApplication的静态run方法，那么，这个方法里面首先要创建一个SpringApplication对象实例，然后调用这个创建好的SpringApplication的实例方法。在SpringApplication实例初始化的时候，它会提前做几件事情：

- 根据classpath里面是否存在某个特征类（org.springframework.web.context.ConfigurableWebApplicationContext）来决定是否应该创建一个为Web应用使用的ApplicationContext类型。
- 使用SpringFactoriesLoader在应用的classpath中查找并加载所有可用的ApplicationContextInitializer。
- 使用SpringFactoriesLoader在应用的classpath中查找并加载所有可用的ApplicationListener。
- 推断并设置main方法的定义类。

2） SpringApplication实例初始化完成并且完成设置后，就开始执行run方法的逻辑了，方法执行伊始，首先遍历执行所有通过SpringFactoriesLoader可以查找到并加载的SpringApplicationRunListener。调用它们的started()方法，告诉这些SpringApplicationRunListener，“嘿，SpringBoot应用要开始执行咯！”。

3） 创建并配置当前Spring Boot应用将要使用的Environment（包括配置要使用的PropertySource以及Profile）。

4） 遍历调用所有SpringApplicationRunListener的environmentPrepared()的方法，告诉他们：“当前SpringBoot应用使用的Environment准备

5） 如果SpringApplication的showBanner属性被设置为true，则打印banner。

6） 根据用户是否明确设置了applicationContextClass类型以及初始化阶段的推断结果，决定该为当前SpringBoot应用创建什么类型的ApplicationContext并创建完成，然后根据条件决定是否添加ShutdownHook，决定是否使用自定义的BeanNameGenerator，决定是否使用自定义的ResourceLoader，当然，最重要的，将之前准备好的Environment设置给创建好的ApplicationContext使用。

7） ApplicationContext创建好之后，SpringApplication会再次借助Spring-FactoriesLoader，查找并加载classpath中所有可用的ApplicationContext-Initializer，然后遍历调用这些ApplicationContextInitializer的initialize（applicationContext）方法来对已经创建好的ApplicationContext进行进一步的处理。

8） 遍历调用所有SpringApplicationRunListener的contextPrepared()方法。

9） 最核心的一步，将之前通过@EnableAutoConfiguration获取的所有配置以及其他形式的IoC容器配置加载到已经准备完毕的ApplicationContext。

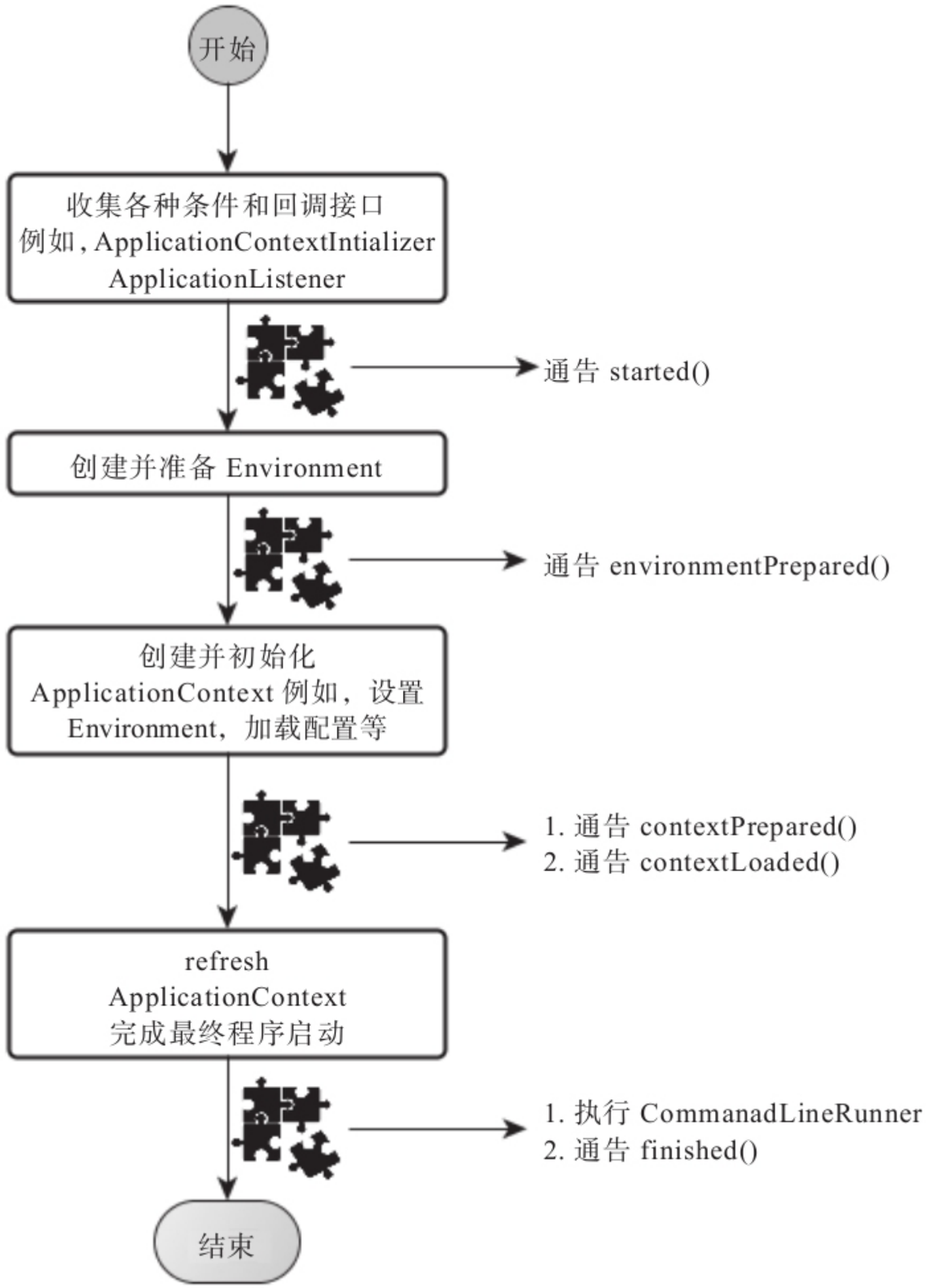
10） 遍历调用所有SpringApplicationRunListener的contextLoaded()方法。

11） 调用ApplicationContext的refresh()方法，完成IoC容器可用的最后一道工序。

12） 查找当前ApplicationContext中是否注册有CommandLineRunner，如果有，则遍历执行它们。

13） 正常情况下，遍历执行SpringApplicationRunListener的finished()方法、（如果整个过程出现异常，则依然调用所有SpringApplicationRunListener的finished()方法，只不过这种情况下会将异常信息一并传入处理）

去除事件通知点后，整个流程如下：


开始
收集各种条件和回调接口
例如，ApplicationContextIntializer
ApplicationListener
通告 started()
创建并准备 Environment
通告 environmentPrepared()
创建并初始化
ApplicationContext 例如，设置
Environment，加载配置等
1. 通告 contextPrepared()
2. 通告 contextLoaded()
refresh
ApplicationContext
完成最终程序启动
1. 执行 CommanadLineRunner
2. 通告 finished()
结束

## 总结

到此，SpringBoot的核心组件完成了基本的解析，综合来看，大部分都是Spring框架背后的一些概念和实践方式，SpringBoot只是在这些概念和实践上对特定的场景事先进行了固化和升华，而也恰恰是这些固化让我们开发基于Sping框架的应用更加方便高效。

想要查看更多Spring Boot干货教程,可前往：Spring Boot干货系列总纲

## 参考

本章节大部分参考了《SpringBoot揭秘快速构建为服务体系》这本书的第三章，个人看过的几本书就感觉这本书介绍原理的章节最为透彻，本章也算这本书最精华的部分。所以我没忍住就分享出来给大家学习。当然主要是我也没有这本书的电子版，无法分享给大家了，深表遗憾(。・__・。)ノ

---

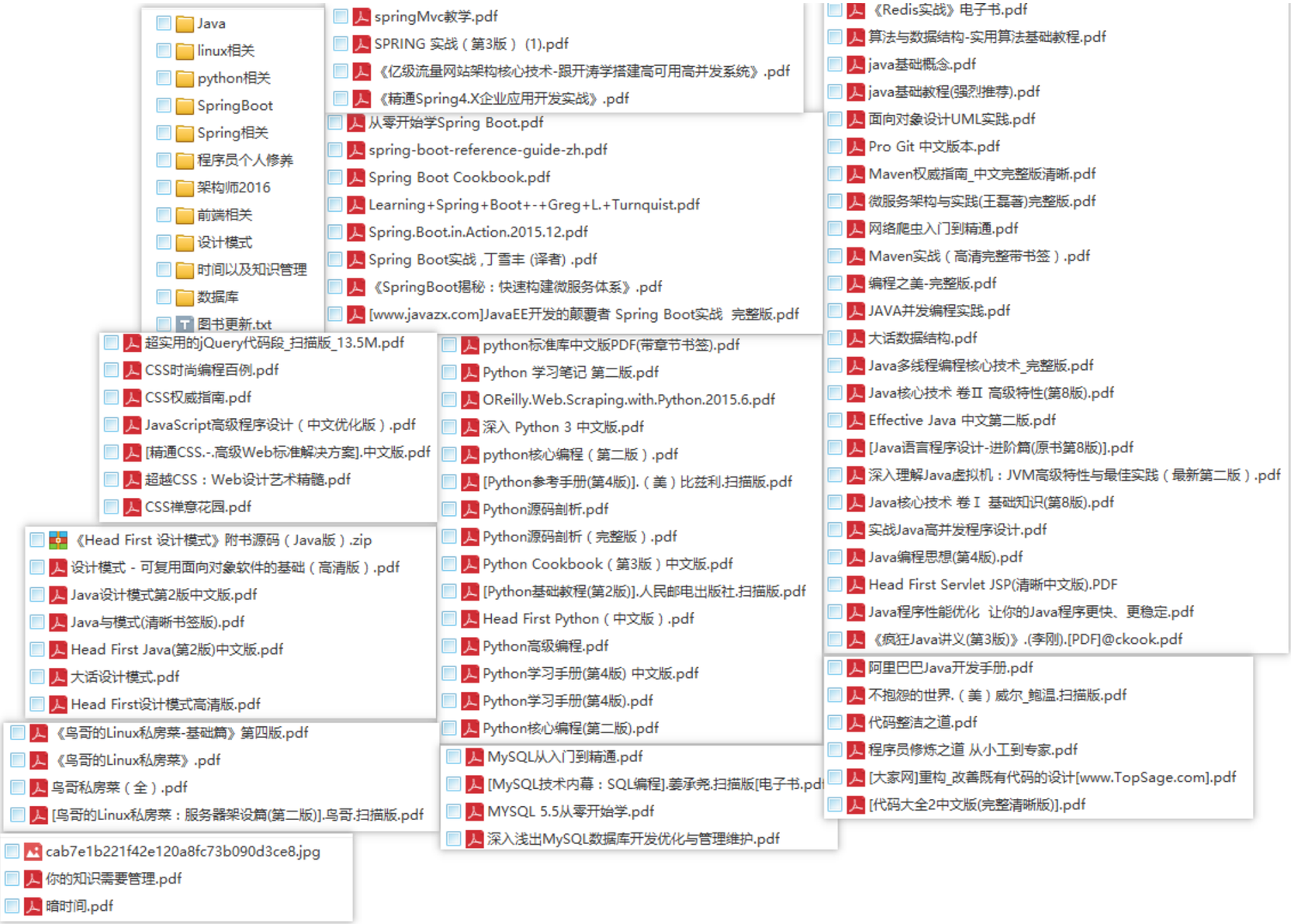
一直觉得自己写的不是技术，而是情怀，一篇篇文章是自己这一路走来的痕迹。靠专业技能的成功是最具可复制性的，希望我的这条路能让你少走弯路，希望我能帮你抹去知识的蒙尘，希望我能帮你理清知识的脉络，希望未来技术之巅上有你也有我,希望大爷你看完打赏点零花钱给我。

订阅博主微信公众号：嘟爷java超神学堂（javaLearn）三大好处：

- 获取最新博主博客更新信息，首发公众号
- 获取大量视频，电子书，精品破解软件资源
- 可以跟博主聊天，欢迎程序媛妹妹来撩我

---

博主最近发起了《嘟爷电子书互惠组》计划，里面包含了《精通Spring4.X企业应用开发实战》相关书籍在内的至少227本Java相关的电子书，也有博主花钱买的电子书。可谓新手必备之物，详情可前往书单末尾查看: Java后端2017书单推荐


Java
linux相关
python相关
SpringBoot
Spring相关
程序员个人修养
架构师2016
前端相关
设计模式
时间以及知识管理
数据库
图书更新.txt
springMvc教学.pdf
SPRING 实战（第3版）(1).pdf
《亿级流量网站架构核心技术-跟开涛学搭建高可用高并发系统》.pdf
《精通Spring4.X企业应用开发实战》.pdf
从零开始学Spring Boot.pdf
spring-boot-reference-guide-zh.pdf
Spring Boot Cookbook.pdf
Learning+Spring+Boot+-+Greg+L.+Turnquist.pdf
Spring.Boot.in.Action.2015.12.pdf
Spring Boot实战 ,丁雪丰 (译者) .pdf
《SpringBoot揭秘：快速构建微服务体系》.pdf
[www.javazx.com]JavaEE开发的颠覆者 Spring Boot实战 完整版.pdf
《Redis实战》电子书.pdf
算法与数据结构-实用算法基础教程.pdf
java基础概念.pdf
java基础教程(强烈推荐).pdf
面向对象设计UML实践.pdf
Pro Git 中文版本.pdf
Maven权威指南_中文完整版清晰.pdf
微服务架构与实践(王磊著)完整版.pdf
网络爬虫入门到精通.pdf
Maven实战（高清完整带书签）.pdf
编程之美-完整版.pdf
JAVA并发编程实践.pdf
大话数据结构.pdf
Java多线程编程核心技术_完整版.pdf
Java核心技术 卷II 高级特性(第8版).pdf
Effective Java 中文第二版.pdf
[Java语言程序设计-进阶篇(原书第8版)].pdf
深入理解Java虚拟机：JVM高级特性与最佳实践（最新第二版）.pdf
Java核心技术 卷I 基础知识(第8版).pdf
实战Java高并发程序设计.pdf
Java编程思想(第4版).pdf
Head First Servlet JSP(清晰中文版).PDF
Java程序性能优化 让你的Java程序更快、更稳定.pdf
《疯狂Java讲义(第3版)》.(李刚).[PDF]@ckook.pdf
超实用的jQuery代码段_扫描版_13.5M.pdf
CSS时尚编程百例.pdf
CSS权威指南.pdf
JavaScript高级程序设计（中文优化版）.pdf
[精通CSS.-.高级Web标准解决方案].中文版.pdf
超越CSS：Web设计艺术精髓.pdf
CSS禅意花园.pdf
《Head First 设计模式》附书源码（Java版）.zip
设计模式 - 可复用面向对象软件的基础（高清版）.pdf
Java设计模式第2版中文版.pdf
Java与模式(清晰书签版).pdf
Head First Java(第2版)中文版.pdf
大话设计模式.pdf
Head First设计模式高清版.pdf
《鸟哥的Linux私房菜-基础篇》第四版.pdf
《鸟哥的Linux私房菜》.pdf
鸟哥私房菜（全）.pdf
[鸟哥的Linux私房菜：服务器架设篇(第二版)].鸟哥.扫描版.pdf
cab7e1b221f42e120a8fc73b090d3ce8.jpg
你的知识需要管理.pdf
暗时间.pdf
python标准库中文版PDF(带章节书签).pdf
Python 学习笔记 第二版.pdf
OReilly.Web.Scraping.with.Python.2015.6.pdf
深入 Python 3 中文版.pdf
python核心编程（第二版）.pdf
[Python参考手册(第4版)].（美）比兹利.扫描版.pdf
Python源码剖析.pdf
Python源码剖析（完整版）.pdf
Python Cookbook（第3版）中文版.pdf
[Python基础教程(第2版)].人民邮电出版社.扫描版.pdf
Head First Python（中文版）.pdf
Python高级编程.pdf
Python学习手册(第4版) 中文版.pdf
Python学习手册(第4版).pdf
Python核心编程(第二版).pdf
MySQL从入门到精通.pdf
[MySQL技术内幕：SQL编程].姜承尧.扫描版[电子书.pdf
MYSQL 5.5从零开始学.pdf
深入浅出MySQL数据库开发优化与管理维护.pdf
阿里巴巴Java开发手册.pdf
不抱怨的世界.（美）威尔_鲍温.扫描版.pdf
代码整洁之道.pdf
程序员修炼之道 从小工到专家.pdf
[大家网]重构_改善既有代码的设计[www.TopSage.com].pdf
[代码大全2中文版(完整清晰版)].pdf


觉得还不错，不妨扫扫右上角的支付宝红包~ 你好我也好，二维码是我媳妇的。感谢支持~



← Spring Boot干货系列：（四）开发Web应用之Thymeleaf篇
Spring Boot干货系列：（二）配置文件解析 →


登录

来说两句吧...

评论

19 人参与, 11 条评论

最新评论

曹思月 [上海市网友] 2017年11月17日 3:46

回复

卖女孩de [山东省青岛市网友] 2017年10月18日 22:18

写的特别好,为博主点赞!

回复

E时行乐 2017年10月8日 3:37

看过书啦，最重要的一点是，书里面启动步骤第9步，最核心的部分，是最复杂的，但是书里面却是说的最少的一步。

回复

行者无疆 [北京市网友] 2017年9月27日 2:04

讲的很到位，对于初学者来说比较合适！

回复 1

K 2017年8月4日 3:19

非常清晰明朗，尤其是spring-boot的启动流程，和其他地方所谓讲解实为拷代码的野路子相比好太多了

回复

来自嘟嘟独立博客wap版

Tom猫 2017年7月26日 22:31

博主，讲@EnableAutoConfiguration的语言有些晦涩，不是很明白。这个注解是修饰哪个类的，能不能给个例子?
--
所以，@EnableAutoConfiguration自动配置的魔法骑士就变成了：从classpath中搜寻所有的META-INF/spring.factories配置文件，并将其中org.springframework.boot.autoconfigure.EnableutoConfiguration对应的配置项通过反射（Java Refletion）实例化为对应的标注了@Configuration的JavaConfig形式的IoC容器配置类，然后汇总为一个并加载到IoC容器。
--

回复

诛笑靥 [广东省广州市网友] 2017年7月17日 2:42

不得不登录来表扬一下你，写的真不错，目前正在积累一些流行框架知识

回复

六个六 [广东省深圳市网友] 2017年6月26日 4:51

写的不错!赞！！

回复

深蔚蓝 [北京市网友] 2017年4月28日 2:44

org.springframework.boot.autoconfigure.EnableutoConfiguration你手误最后单词少写了A
博主你写的是：（org.springframework.web.context.ConfigurableWebApplicationContext）来决定是否应该创建一个为Web应用使用的ApplicationContext类型。
我走源码发现run（）的类型是ConfigurableApplicationContext。我的理解哪里出了问题吗?

回复 2

来自嘟嘟独立博客wap版

嘟嘟MD [福建省福州市网友] 2017年4月25日 2:01

王爵nice [上海市网友] 1

为博主点赞，写的不错!

谢谢

回复 3

王爵nice [上海市网友] 2017年4月24日 22:13

为博主点赞，写的不错!

回复 2

嘟嘟独立博客正在使用畅言

热评话题

如何制作微课系列——（第一天）微课选题怎么玩？ | 嘟嘟独立博客
Spring Boot干货系列：（十）开发常用的热部署方式汇总 | 嘟嘟独...
Spring Boot干货系列：（十一）数据存储篇-Spring Boot整合Myba...


精品软件推荐—百度云2018最新限速破解软件 | 嘟嘟独立博客

使用高级搜索指令提高搜索效率（百度，谷歌） | 嘟嘟独立博客

springMVC干货系列：从零搭建springMVC mybatis（一）：mave...